Article
Guillaume Rongier · Août 29, 2022 9m de lecture

Découvrir Django, partie 2

Dans la première partie, j'ai montré comment démarrer un nouveau projet sur Django, ainsi que définir de nouveaux modèles et ajouter des modèles existants.

Cette fois, je vous présenterai un panneau d'administration, qui est disponible dès le départ, et comment il peut être utile.

Remarque importante : ne vous attendez pas à ce que si vous essayez de répéter les actions de cet article, cela fonctionnera pour vous, ce n'est pas le cas. Au cours de l'article, j'ai dû faire quelques corrections dans le projet django-iris, et même dans le pilote DB-API fait par InterSystems pour corriger certains problèmes là aussi, et je pense que ce pilote est encore en développement, et nous aurons un pilote plus stable dans le futur. Considérons que cet article ne fait qu'expliquer comment cela pourrait être si nous avions tous fait.

Revenons à notre code et voyons ce que nous avons dans urls.py, qui est le principal point d'entrée de toutes les requêtes web.

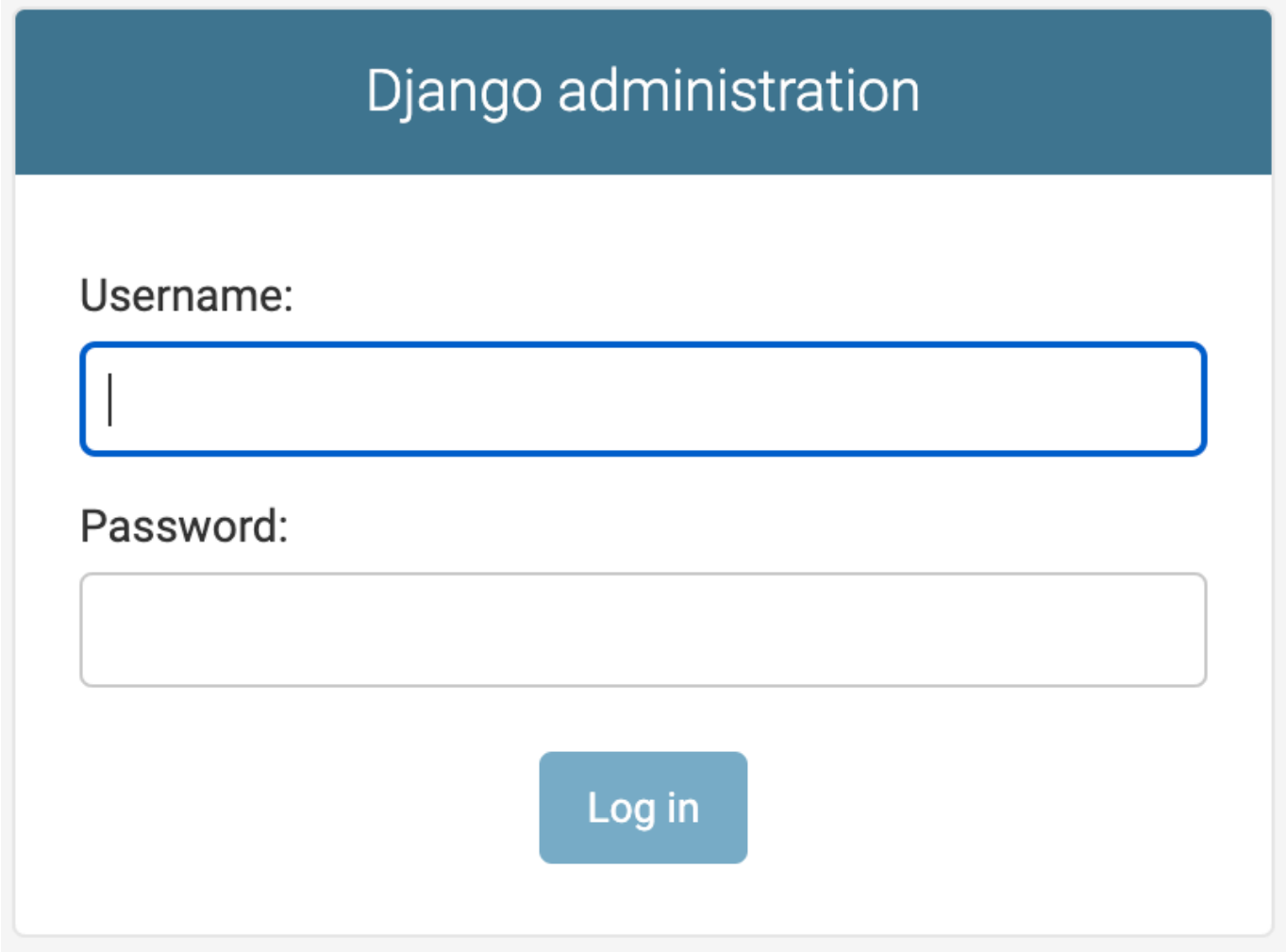
```
"""main URL Configuration

La liste `urlpatterns` dirige les URLs vers les visualisations. Pour plus d'informati
ons, veuillez consulter :
    https://docs.djangoproject.com/en/4.0/topics/http/urls/
    Exemples :
    Visualisations des fonctions
    1. Ajoutez un import :  from my_app import views
    2. Ajouter une URL à urlpatterns :  path('', views.home, name='home')
Visualisations basées sur des classes
    1. Ajoutez un import :  from other_app.views import Home
    2. Ajouter une URL à urlpatterns :  path('', Home.as_view(), name='home')
En incluant une autre URLconf
    1. Importez la fonction include() : from django.urls import include, path
    2. Ajoutez une URL à urlpatterns :  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path

urlpatterns = [
    path('admin/', admin.site.urls),
]
```

Comme vous pouvez le constater, l'URL ***admin*** y est déjà définie

Lançons le serveur de développement, par la commande

```
python manage.py runserver
```

Si vous passez par l'URL http://localhost:8000/admin, vous trouverez le formulaire de connexion à l'administration de Django



Pour entrer ici, nous avons besoin d'un utilisateur, et nous pouvons le créer avec cette commande

```
$ python manage.py createsuperuser
Nom d'utilisateur (laissez vide pour utiliser 'daimor') : admin
Adresse e-mail : admin@example.com
Mot de passe :
Mot de passe ( de nouveau) :
Le mot de passe est trop similaire au nom d'utilisateur.
Ce mot de passe est trop court. Il doit contenir au moins 8 caractères.
Ce mot de passe est trop courant.
Contourner la validation du mot de passe et créer un utilisateur quand même ? [O/N] :
 O
Le superutilisateur a été créé avec succès.
```

Nous pouvons maintenant utiliser ce nom d'utilisateur et ce mot de passe. Il est assez vide pour le moment, mais il donne déjà accès aux groupes et aux utilisateurs.

Plus de données

Précédemment j'ai déjà installé le paquet post-and-tags avec zpm, vous pouvez le faire aussi

```
zpm "install posts-and-tags"
```

Maintenant nous pouvons obtenir les modèles pour tous les tableaux (*community.post, community.comment, community.tag*) installés par ce paquet

```
$ python manage.py inspectdb community.post community.comment community.tag > main/mo
dels.py
```

Le fichier sera un peu long, donc, je l'ai mis dans un spoiler

main/models.py

Le tableau de bord d'administration en Django peut être étendu par le développeur. Il est ainsi possible d'ajouter des tableaux supplémentaires. Pour ce faire, nous devons ajouter un nouveau fichier nommé **main/admin.py.** J'ai ajouté quelques commentaires dans le code, pour expliquer certaines lignes.

```
from django.contrib import admin

# Importation de nos modèles communautaires pour nos tableaux dans IRIS
from .models import (CommunityPost, CommunityComment, CommunityTag)

# classe de registre qui remplace le comportement par défaut du modèle CommunityPost
@admin.register(CommunityPost)
class CommunityPostAdmin(admin.ModelAdmin):
  # liste des propriétés à afficher dans le tableau view
  list_display = ('posttype', 'name', 'publisheddate')

# liste des propriétés pour lesquelles il faut afficher un filtre à la droite du tabl
eau
  list_filter = ('posttype', 'lang', 'published')
  # ordre par défaut, c'est-à-
dire à partir de la date la plus récente de PublishedDate
  ordering = ['-publisheddate', ]

@admin.register(CommunityComment)
class CommunityCommentAdmin(admin.ModelAdmin):

# seuls ces deux champs apparaissent, ( le post est numérique par id dans le tableau
post)
  list_display = ('post', 'created')
  # classées par date de création
  ordering = ['-created', ]

@admin.register(CommunityTag)
class CommunityTagAdmin(admin.ModelAdmin):
  # pas tellement à montrer
  list_display = ('name', )
```

Portail d'extension

Retournons à notre page d'administration de Django, et examinons les nouveaux éléments qui y ont été ajoutés

Sur le côté droit, vous verrez le panneau de filtrage, et ce qui est important, c'est qu'il contient toutes les valeurs

possibles dans des champs particuliers et qu'il les affiche.

Malheureusement, InterSystems SQL ne permet pas d'utiliser les fonctions LIMIT, OFFSET, attendues d'une manière ou d'une autre par Django. Par ailleurs, Django ne permet pas d'utiliser la fonction TOP. Ainsi, la pagination sera affichée ici mais ne fonctionnera pas. Et il n'y a aucun moyen de la faire fonctionner pour le moment, et je ne pense pas que cela fonctionnera un jour, malheureusement.

Vous pouvez même plonger dans l'objet, et Django affichera le formulaire, avec les types de champs corrects. (*remarque : Cet ensemble de données ne contient pas de données dans le champ Texte*)

Objet du commentaire

Problèmes avec les licences à prévoir sur Community Edition

#Python #InterSystems IRIS